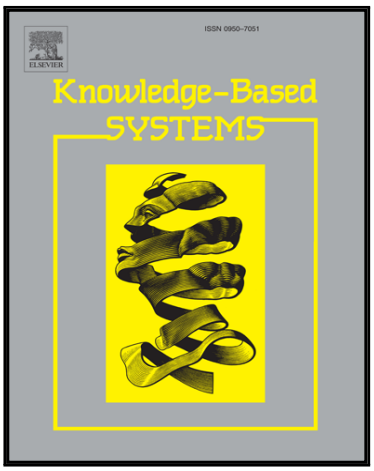


When Ensemble Learning Meets Deep Learning: a New Deep Support Vector Machine for Classification

Zhiquan Qi, Bo Wang, Yingjie Tian, Peng Zhang

PII: S0950-7051(16)30160-5
DOI: 10.1016/j.knosys.2016.05.055
Reference: KNOSYS 3552

To appear in: *Knowledge-Based Systems*

Received date: 10 November 2015
Revised date: 9 March 2016
Accepted date: 28 May 2016

Please cite this article as: Zhiquan Qi, Bo Wang, Yingjie Tian, Peng Zhang, When Ensemble Learning Meets Deep Learning: a New Deep Support Vector Machine for Classification, *Knowledge-Based Systems* (2016), doi: 10.1016/j.knosys.2016.05.055

Highlights

- Focusing on the DL research based on Support Vector Machine (SVM)
- First give an Ex-Adaboost learning strategy
- Proposing a new Deep Support Vector Machine (called DeepSVM)
- Using Ex-Adaboost to not only select SVMs with the minimal error rate and the biggest diversity, but also give each feature's weight for the new training data
- Obtaining a new set of deep features that can been classified more easily
- Entering training data represented by these new features into a standard SVM

When Ensemble Learning Meets Deep Learning: a New Deep Support Vector Machine for Classification

Zhiquan Qi[a,b], Bo Wang[a,b], Yingjie Tian[a,b,*], Peng Zhang[c]

[a] *Research Center on Fictitious Economy and Data Science of Chinese Academy of Sciences, Beijing, China, 100190*
[b] *Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing, China, 100190*
[c] *University of Technology, Sydney, NSW 2007, Australia*

Abstract

Recently, Deep Learning (DL) method has received a significant breakthrough in the data representation, whose success mainly depends on its deep structure. In this paper, we focus on the DL research based on Support Vector Machine (SVM), and first present an Ex-Adaboost learning strategy, and then propose a new Deep Support Vector Machine (called DeepSVM). Unlike other DL algorithms based on SVM, in each layer, Ex-Adaboost is applied to not only select SVMs with the minimal error rate and the highest diversity, but also to produce the weight for each feature. In this way, new training data is obtained. By stacking these SVMs into multiple layers following the same way, we finally acquire a new set of deep features that can greatly boost the classification performance. In the end, the training data represented by these new features is regarded as the input for a standard SVM classifier. In the experimental part, we offer these answers to the following questions: 1) is the deep structure of DeepSVM really useful for classification problem? 2) Does Ex-Adaboost work, and is it helpful for further improving on DeepSVM's performance with respect to the deep structure? 3) How much improvement in classification accuracy of DeepSVM, compared with other exist algorithms?

Keywords: pattern recognition, deep architectures, support vector machine

[*] **Corresponding Author:** Yingjie Tian; Email: tyj@ucas.ac.cn; Tel: +86-10-82680997; Fax: +86-10-82680997.

1. Introduction

For the classification problem, the performance of learning algorithms extremely depends on the nature of data representation (Bengio et al., 2013). In 2006, Geoff Hinton reported a significant breakthrough in the feature extraction, and firstly proposed the concept of "Deep Learning (DL)" (Hinton and Salakhutdinov, 2006; Hinton et al., 2006). After then, DL has generated considerable recent research interest in various fields (Ciresan et al., 2010; Socher et al., 2011; Goodfellow et al., 2009; Bengio, 2012). Essentially, as a feature extracting method, DL attempts to obtain the high-level feature abstractions by learning multiple feature architectures in the training process. Each iteration of DL is an unsupervised feature learning process, and the combination of multiple layers can produce a deep supervised predictor (Bengio et al., 2013).

Deep Belief Networks (DBN) based on Restricted Boltzmann Machines (RBMs) is a representative algorithm of DL (Hinton et al., 2006), which is comprised of multiple layers of hidden units. There are connections between the layers but none among units within each layer (Hinton, 2009). First, DBN implements to train data in an unsupervised learning way. In this stage, DBN learns the common features of inputs as much as possible. Second, DBN can be further optimized in a supervised way, and then constructs the corresponding model for classification or other pattern recognition task. Convolutional Neural Networks (CNN) is another example of DL (LeCun et al., 1998; Fukushima, 1980; Le Cun et al., 1990; Jarrett et al., 2009; Simard et al., 2003), which greatly successes in digit recognition field (Jarrett et al., 2009). CNN is inspired by the visual system's structure, and imitates the visual system to extract and recombine the orientations information. In addition, CNN is also a multiple layers neural networks, and each layer contains several two-dimensional planes that are composed by many neurons. One principal advantage of CNN is that the weight in each convolutional layers is shared. In other words, the neurons in each two-dimensional plane uses the same filter, which greatly reduces tunable parameters and the computation complexity (Arnold et al., 2011; Bengio, 2009). Auto Encoder (AE) is also stacked to train a deep architectures in a greedy layer-wise manner (Bourlard and Kamp, 1988; Hinton, 1989; Hinton and Zemel, 1994; Japkowicz et al., 2000; Rumelhart et al., 1988; Schwenk and Milgram, 1995; Arnold et al., 2011). For a Neural Networks (NN) system, suppose that the output itself is the input data. By adjusting the weight of each layer, we will be able

to obtain the different data representations for the original data. In other words, AE is an NN of reconstructing the input data, which is composed of encoder and decoder. In addition, other DL methods can be found in Arnold et al. (2011); Kim et al. (2015); Carrasco et al. (2015).

2. Related work and Motivation

2.1. Related work

From the literatures above, we can tell that there are various methods to realize the deep architecture of DL. The motivation of this paper focuses on the DL research based on Support Vector Machine (SVM) (Vapnik, 1995, 2006; N and Deka, 2014; Czibula et al., 2014; Azamathulla and Wu, 2011; Ekici, 2012). In the early 90s, Vapnik et al. proposed the famous SVM method, which gradually became one of the most popular machine learning methods in the past few decades. The remarkable success of SVMs due to three key elements: the principle of maximum margin, dual theory, and kernel trick. The standard Support Vector Classification (SVC) maximizes the margin based on structural risk minimization (SRM). Dual theory makes the introduction of kernel function possible. Then, the kernel trick is applied to adapt nonlinear cases (Tian et al., 2013).

How to apply the nature of SVM to further improve the performance of DL is a very valuable research. Fortunately, huge progress has been obtained in this direction nowadays. Sangwook Kim et al. introduced a new family of positive-definite kernel functions to mimic the computation in large, multi-layer neural nets. These kernel functions can be both used in shallow architectures and called Multi-layer Kernel Machines (MKMs) in deep architectures. They evaluated SVMs and MKMs in the experiments on multi-class classification, and showed the advantages of deep architectures (Cho and Saul, 2009, 2010). In addition, Yichuan Tang replaced the soft-max layer of CNN with a linear SVM. The learning mechanism minimized a margin-based loss instead of the cross-entropy loss of CNN. Compared with other methods, the reported results demonstrated that their method could offered significant superiority on popular deep learning datasets, such as MNIST, CIFAR-10, and the ICML 2013 Representation Learning Workshop's face expression recognition challenge (Tang, 2013). On the other hand, Azizi Abdullah et al. proposed a new deep architecture called D-SVM. D-SVM was firstly trained by the traditional way and then applied the kernel activations of support vectors as inputs for the next layer. Furthermore, they combined

different descriptors in an ensemble of deep SVMs, and experimental results illustrated that D-SVM with the product rule performed significantly better than a standard SVM (Abdullah et al., 2009). Furthermore, Sangwook Kim et al. proposed a deep network with SVMs. By stacking SVMs, they extracted features with support vectors which maximized the margin performance. Experimental results testing on Wisconsin Breast Cancer Database also verified its perfect behaviour on every layer. Higher layer SVMs behaved better generalization performance (Kim et al., 2013). M.A. Wiering et al. proposed a deep SVM for regression problems (called MLSVM (Multi-Layer Support Vector Machine)). The system was trained by a simple gradient ascent learning rule on a min-max formulation of the optimization problem. A two-layer DSVM was compared to the regular SVM on ten regression datasets and the results supported that the DSVM outperformed the SVM (Wiering et al., 2013a,b).

2.2. Motivation and contribution

It is true that the methods mentioned above have been successfully implemented DL structure based SVM, but there are still many shortcomings and improvable issues. For example, MKMs uses the arc-cosine kernel functions to realize the deep kernel-based architectures, but the kernel selection is still a challenging problem, which greatly affects the quality of data representation. For D-SVM, how to deicide the parametric-type issues, such as kernel functions, the number of SVMs, weights of features is still unanswered and problematic. Furthermore, MLSVM is trained by a simple gradient ascent learning rule on a min-max formulation of the optimization problem, whose structure is similar to a neural network, in other words, different with the deep structure. As a result, when encountering many parameters of each layers, this method is prone to fall into a local optimum.

In this paper, inspired by the ensemble learning (Freund and Schapire, 1999), we proposed a new Deep Support Vector Machine (called DeepSVM). For each layer, we firstly design an Ex-Adaboost learning strategy to select SVMs with the minimal error rate and the highest diversity. Then, according to the classification performance, we determine each feature's weight for the training data of the next layer. As the number of layers increases, we can guarantee to obtain better data representation thanks to Ex-Adaboost and the deep structure. Finally, standard SVM is applied to train the deep structure transformed data sets to obtain the final classifier.

DeepSVM has the following advantages: 1) different with the traditional deep structure, each layer of DeepSVM is trained by supervised learning. In other words, DeepSVM generates new training data in each layer under the supervision of labels, which facilitates the final classification. 2) Many parameters (such as kernel functions, the number of SVMs, weights of features) can be predefined by Ex-Adaboost. All experiments report that this is a good parameters learning strategy.

In summary, our paper makes two main contributions. First, we develop a new Deep Support Vector Machine. Second, in the experimental part, we try to give answers to a series of problems such as how important of deep structure for SVMs, evaluating the effectiveness of Ex-Adaboost and the performance comparison of DeepSVM with other algorithms.

The remaining content is organized as follows. In Section 3, we methodically introduce our algorithm: DeepSVM. In Section 4, we perform experiments of DeepSVM on various data sets. We will conclude our work and depict outlook in Section 5.

3. DeepSVM

In this section, we focus on the detailed description of the proposed algorithm. Firstly, a classification problem with the training data can be expressed as follows:

$$T = \{(x_1, y_1), \cdots, (x_l, y_l)\} \in (\Re^N \times \mathcal{Y})^l, \tag{1}$$

where $x_i \in \Re^N, y_i \in \mathcal{Y} = \{1, -1\}, i = 1, \cdots, l$. To meet our mechanism, the architecture of DeepSVM is simply illustrated in Fig. 1.

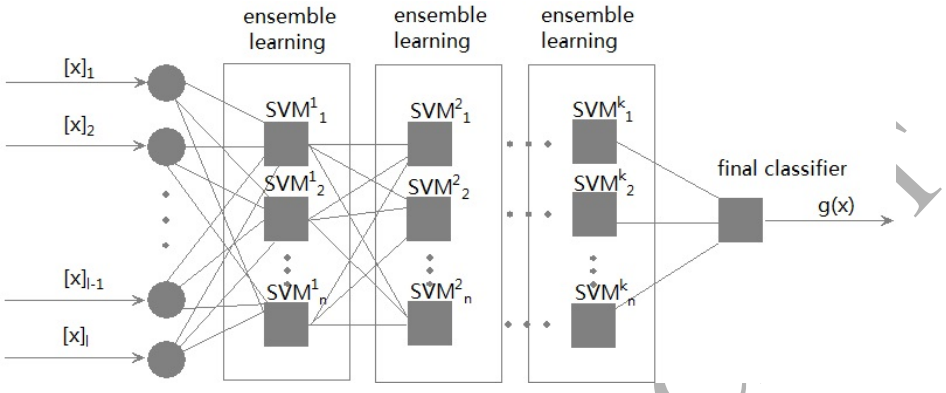


Figure 1: Description for DeepSVM

From Fig.1, we can tell that the multi-layer architecture contains an input layer of data, hidden layers and an output layer. Then, in the k-th layers, there are n SVMs, which are functioned the new features translation for the next layer. Here, how to measure the importance of each feature and how to select the optimal classifiers for the k-th layer are two key issues, which will directly influence the performance of the model. As a main contribution of this paper, we employ an ensemble method (Ex-Adaboost) to supervise the adjustment of kernel parameters and weight coefficients for each layer. In detail, for each layer learning, both the error rate ε_t and P-Value's correlation are considered to tune each weak classifier's weight, which endows the new training data more different and representative features. With the increase of layers, according to deep learning' mechanism, we can obtain better data expression for final classification task, which is a principle difference compared to traditional shallow learning algorithms.

Due to the output of each layer is treated as the new training data for next layer, the k-th layer training data can be expressed as (2)

$$\begin{aligned} T^k = \{&([f_1^{k-1}(f_1^{k-2}\cdots f_1^1)(x_1),\cdots,f_n^{k-1}(f_n^{k-2}\cdots f_n^1)(x_1)],y_1),\cdots, \\ &([f_1^{k-1}(f_1^{k-2}\cdots f_1^1)(x_l),\cdots,f_n^{k-1}(f_n^{k-2}\cdots f_n^1)(x_l)],y_l)\} \in (\Re^n\times\mathcal{Y})^l \end{aligned} \tag{2}$$

Finally, SVM is applied to construct the decision function by training the extracted feature vector in hidden layers. In the following part, we will describe how to use Ex-Adaboost method to extract features which are obtained by hidden layers.

3.1. Extract features via Adaboost (Ex-Adaboost)

In this section, we will utilize Adaboost method to extract new features and construct new training data for each layer. However, standard Adaboost cannot be directly applied to feature extraction of hidden layers. On the other hand, our aim is to find feature expressions as diverse as possible. Thus, the objective function of Adaboost needs to maximize the irrelevance among various features. Here we employ Pearon Value (P-Value) to compute the correlation coefficient r (Stigler, 1989):

$$r = R(X,Y) = \frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum(X-\bar{X})^2(Y-\bar{Y})^2}}, \tag{3}$$

where X, Y represents two different random variable, and $\bar{X}$, $\bar{Y}$ are mean values of X and Y, respectively,. Finally, the new Adaboost method (called Ex-Adaboost) can be expressed as **Algorithm 1**.

Algorithm 1 Ex-Adaboost

Input: Training set in the k-th $\{(x_1^k, y_1), \cdots, (x_l^k, y_l)\} \in (\Re^n \times \mathcal{Y})^l$.
Initialize: The weight of samples: $w_i^1 = 1/l.i = 1, \cdots, l$.
Loop: For $t = 1, \cdots, n$:

- By selecting kernel parameters, obtain the error measurement ϵ_t: $\epsilon_t = a * \frac{1}{r_t} + (1-a) * \varepsilon_t$, where r_t can be written as: $r_t = \min_{t'=1,2,\cdots,t-1} R(f_{t'}(x^k), f_t(x^k))$, and $\varepsilon_t = \sum_{i=1}^{l} w_i^t I[y_i \neq h_t(x_i)]$.
- Set weight of SVM$_t$ $h_t : \alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
- Update training data weights: $w_i^{t+1} = \frac{w_i^t \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, where Z_t is a normalization constant, and $\sum_{i=1}^{l} w_i^{t+1} = 1$.

Output: Obtain new training data: $\{([\alpha_1 f_1^k(x_1^k)/M_1, \cdots, \alpha_n f_n^k(x_1^k)/M_n], y_1), \cdots, ([\alpha_1 f_1^k(x_1^k)/M_1, \cdots, \alpha_n f_n^k(x_l^k)/M_n], y_l)\}$, M_t is the normalization constant of each feature.

In detail, Adaboost employs several classifiers to enhance the performance in classification. The mechanism in Algorithm 1 precisely follows which in original Adaboost. On one hand, Ex-Adaboost algorithm inherits the deep structure with multi-layer. On the other hand, in every individual layer, ensemble learning mechanism is applied to boost the effectiveness of weak classifiers as well as the diversity in multiple classifiers.

For the weight calculation issue, considering the weight α_t of certain classifier h_t, the better the classifier performs, the larger the weight is. Here, the essential difference is when measuring the performance, we consider not only the classification error but also the diversity in multiple classifiers. Particularly, a constant a is chosen to tradeoff the influence these two measurements. Also, w_i^{t+1} represents the data distribution update, which accounts for the calculation for error rate in classification, i.e. ε_{t+1}. Furthermore, we also present the following algorithm (DeepSVM) to construct the final classifier based on the deep feature learned from Ex-Adaboost.

Algorithm 2 DeepSVM

Input: Training set $\{(x_1, y_1), \cdots, (x_l, y_l)\} \in (\Re^N \times \mathcal{Y})^l$.

Initialize: Set the range of C and σ of RBF kernel parameter in SVM, $C = [2^{-8}, \cdots, 2^8], \sigma = [2^{-8}, \cdots, 2^8]$.

Loop: For $k = 1, \cdots, K$:

- Carry out Ex-Adaboost.
- Construct new training data $\{(x_1^{k+1}, y_1), \cdots, (x_l^{k+1}, y_l)\} \in (\Re^n \times \mathcal{Y})^l$.

Output: Obtain the final classifier:

$$f(x) = sgn(\sum_{i=1}^{n} y_i \alpha_i^* (x_i^{K+1} \cdot x) + b^*).$$

Due to the standard SVM can not set the different weight coefficient for each sample, we need to use a converted version of standard SVM for **Algorithm 1** Deng and Tian (2009):

$$\begin{aligned} \max_{\alpha} \quad & \sum_{j=1}^{l} \alpha_j - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j (x_i \cdot x_j) \alpha_i \alpha_j \\ \text{s.t.} \quad & \sum_{i=1}^{l} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C_i, \ i = 1, \cdots, l, \end{aligned} \tag{4}$$

where C_i can be set as the weight coefficient obtained by Ex-Adaboost algorithm.

3.2. Discussion

There are two remarkable characters of deep learning, i.e. pre-training and regularization. Our method focuses on the auto-encoder technique and tries

to integrate ensemble learning into this procedure, which can be viewed as pre-training in deep learning. Particularly, in standard deep learning setting, each layer can be viewed as an specific abstract representation for original data. However, this representation needs to be adjusted in the following two steps, which means feed-forward and back-propagation. These procedures are time-consuming and with high computational complexity. By contrast, our method focuses on the layers individually. Take advantage of ensemble learning, our method only deals with classifiers in the same layer in each iteration, and constructs new feature extraction for original data. Later, this new data set is exploited as the input for the next layer. In this way, our procedure only tunes the weights in the feed-forward step, and there is no back-propagation in our algorithm. In a word, every layer is independent. In every layer, Adaboost is employed to update the weights of classifiers in the same layer with respect to the evaluation of errors in classification. Also, it updates the distribution of training data according to these errors, which is a key principle in ensemble learning, i.e. different classifiers in the same layer are based on different data distributions.

Taking the outputs of SVMs in each layer as the sample inputs in new feature space is the crucial character of the proposed multi-layer learning algorithms. However, how to measure the importance of each feature and how to select the optimal classifiers for each layer are two key problems, which have rarely been studied in former literatures. In this paper, we implement a simple but efficient Ex-Adaboost learning tactics to meet these two challenges. For each layer learning, by means of changing each sample's weight and introducing the P-Value's correlation analysis, Ex-Adaboost can produce different weak classifiers, which harnesses satisfactory features to comprise the new training data. In addition, with these more different and representative features, we realise effective data description. At the same time, according to the error rate ε_t for each weak classifier, we can obtain different weights for each feature, which will be helpful for next layer's feature expression. With the increase of layers, according to deep learning mechanism, we can obtain better data expression for classification, which is a main difference between the proposed algorithm and traditional shallow learning algorithms. However, the absence of strictly theoretical analysis for our Deep SVM should be noticed, compared to other deep learning algorithms. To address this issue, in the next section, we will report abundant experimental results to demonstrate the justifiability of our algorithm and evaluate its performance.

4. Experiments

The experimental environment is: Intel Core i7-2600 CPU, 4 GB memory. LIBSVM (Chang and Lin, 2011) is applied to solve the standard SVM. All codes are executed in the Matlab 6.0. The methods selected to offer comparison results include: SVM (Vapnik, 2006), DeepSVM, stacked SVM, LMSVM (Wiering et al., 2013a), MKMS (Cho and Saul, 2009) and Deep Neural Netwak (DNN)[1].

4.1. DeepSVM's performance analysis

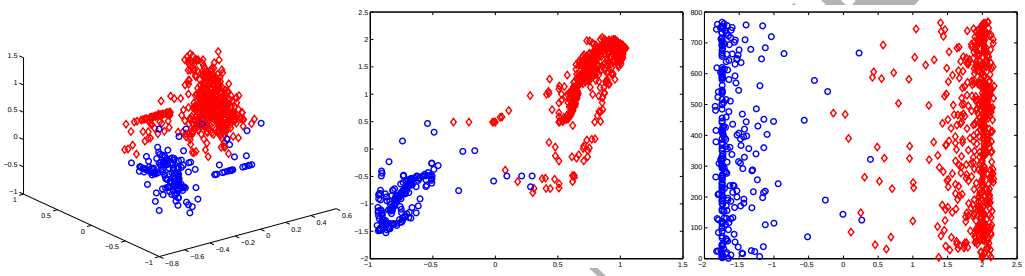
In the first experiment, we will try to answer these following questions: 1) is the deep structure of DeepSVM really useful for classification problem? 2) Does Ex-Adaboost work, and is it helpful for further improving on DeepSVM's performance with respect to the deep structure? To this end, we use 8 UCI data sets in our experiment [2]: diabetes, ala, breast−cancer, australian, german_numer, ionosphere, splice, w1a (Murphy and Aha., 1992). Here, since we don't need to test DeepSVM's absolute performance, only three layers structure is chosen to explain the methodology. In order to intuitively visualize our algorithms, we set $n = 3, 2, 1$ in the first, second, third layer, respectively. In the meantime, we slightly revise DeepSVM by removing Ex-Adaboost tactics (called stacked SVM) to further manifest the importance of ensemble process. Here, the testing accuracies are computed with standard 10-fold cross validation. Besides, C and RBF kernel parameter are all chosen from the set $\{2^i | i = -8, \cdots, 8\}$ with respect to 10-fold cross validation results on the tuning set comprising of random selection from the training data with 10% proportion. Once all the parameters are fixed, the tuning set was returned to the training set to build the final decision function. For stacked SVM, we select three most optimal sets of parameters to perform SVMs in each layer.

We display Fig. 2 to demonstrate the diabetes's result. For each row, we can intuitively tell that the training data can be classified increasingly easily with the increase of layers. For each column, we can also notice that the overlapping samples in DeepSVM are much less than that of stacked

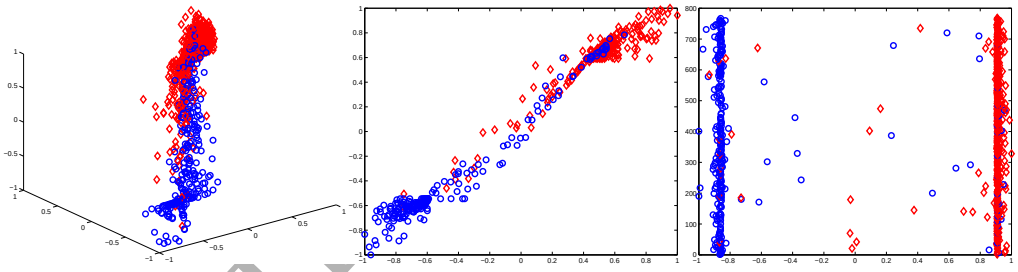
[1] https://www.mathworks.com/matlabcentral/fileexchange/42853-deep-neural-network

[2] These data sets in LIBSVM format are available at http://www.csie.ntu.edu.tw/ cjlin/libsvm/index.html

SVM. In other words, the new training data generated by DeepSVM is easier to be classified than that of stacked SVM. This concludes that the deep structure is indeed helpful in improving our algorithm's performance. Finally, the accuracy of DeepSVM is 79.68%, which is higher than stacked SVM's 78.38%. Other data sets' results are listed in Table 1. The average accuracy of DeepSVM in all datasets is at least 0.82 percent higher than that of stacked SVM. This observation fully supports that Ex-Adaboost promotes the data representation ability of DeepSVM.



(a) DeepSVM



(b) stacked SVM

Figure 2: DeepSVM and stacked SVM's feature representations in different layers about the diabetes data set. The 1,2 3 column are the results in 1,2,3 layer. Different coordinates denote different features.

In the second experiment, we will try to answer these questions as follows: 1) what is the accuracies of DeepSVM with different numbers of layers under the optimal parameters? 2) is the performance of DeepSVM better than stacked SVM and standard SVM under the optimal parameters? The selected UCI data sets and the method of selecting parameters are both the same as the first experiment. In detail, we perform 15 SVMs to construct a new training data in each layer. The number of the depth of deep structure is set to 4. The final result can been found in Fig. 3 and Table 2.

Table 1: The final 10-fold cross validation's testing accuracies in UCI data sets for the first experiment.

datasets	diabetes	ala	breast−cancer	australian
stacked SVM	78.38	83.54	98.24	85.46
DeepSVM	79.68	84.66	98.53	86.15
dataset	german_numer	ionosphere	splice	w1a
stacked SVM	71.40	88.56	85.91	97.09
DeepSVM	71.80	89.84	86.37	97.25

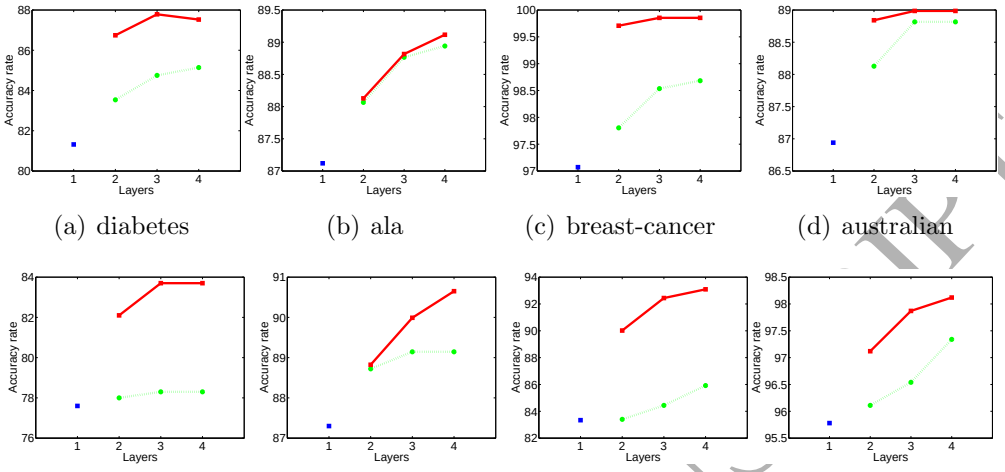
Table 2: The final 10-fold cross validations testing accuracies in UCI data sets for the second experiment

Datasets	SVM Accuracy	stacked SVM Accuracy	MLSVM Accuracy	MKMS Accuracy	DNN Accuracy	DeepSVM Accuracy
diabetes	81.32±1.22	85.14±2.13	86.48±1.24	86.34±3.28	86.96±1.78	**87.52±2.30**
ala	87.12±0.89	88.94±2.38	87.56±3.90	86.88±2.45	88.65±3.08	**89.12±1.97**
breast−cancer	97.07±1.08	98.68±1.34	98.66±2.12	97.84±2.08	99.21±1.07	**99.85±3.11**
australian	86.94±1.56	88.81±2.11	87.33±3.11	87.23±1.80	87.78±1.75	**88.98±2.09**
german_numer	77.60±1.21	78.30±2.39	81.12±1.89	82.98±3.21	**84.12±0.99**	83.70±1.33
ionosphere	87.30±1.24	89.14±1.26	88.23±3.24	88.67±3.45	87.86±2.87	**90.65±1.44**
splice	83.34±0.76	85.92±1.26	89.02±2.21	92.33±1.56	92.43±1.92	**93.09±2.33**
w1a	95.78±1.34	97.34±1.69	96.78±2.11	97.34±1.28	**98.18±1.05**	98.12±2.06

From Fig. 3 and Table 2, we can tell that the performance of DeepSVM in the back layer is better than that of front layer in most cases (Only in the case of the diabetes, the accuracy of DeepSVM in the fourth layer is slightly lower than that in the third layer.). This demonstrates the deep structure is somehow effective in the case of optimal parameters. In addition, DeepSVM, stacked SVM, MLSVM and DNN outperform SVM (SVM can be regarded as one layer and one classifier constructed DeepSVM). From this point of view, deep structure is a reasonable reformation for DeepSVM and stacked SVM. Meanwhile, DeepSVM outperforms stacked SVM, MLSVM and DNN in most of cases, which shows that Ex-Adaboost (ensemble) is an important process for DeepSVM.

4.2. Horizontal Comparison

In this subsection, we use image data sets to evaluate SVM, DeepSVM and other SVM algorithms with deep structure: MLSVM and MKMS Cho and Saul (2009); Wiering et al. (2013a). Because we aim to compare the performance between DeepSVM and other algorithms, all experiments are



(a) diabetes (b) ala (c) breast-cancer (d) australian

(e) german-numer (f) ionosphere (g) splice (h) w1a

Figure 3: The 10-fold cross validation's testing accuracies in different layers of UCI data sets. Blue point: SVM; green point: stacked SVM; red point: DeepSVM.

carried out on raw pixel features, in order to get rid of the influence of data preprocess.

1) ABCDETC dataset (Weston et al., 2006): there are 78 classes in the data set, containing 19,646 images. These categories are respectively listed as follows: lowercase letters('a-z'), uppercase letters ('A-Z'), digits ('0- 9') and other symbols (', . : ; ! ? + - = / $ % ( ) @ " '). Those symbols were finished in pen by 51 subjects, that wrote 5 versions for each symbol on a single gridded sheet, and then saved as 100×100 binary images. To improve the computing speed, we shrunk the original images into 30×30 pixels by the bilinear interpolation method. The ',' and '.', '/' and ')', '!' and '?', 'a' and '@' were taken as the labeled data respectively. The size of training data was 140, and other 140 samples were left as testing data. Each experiment was repeated 10 times. In the end, Fig. 4 displayed the average results.

In the experiment of ',' and '.' , we can realise that DeepSVM outperforms than other three algorithms, and MKMS outperforms MLSVM and standard SVM. The result of 'a' and '@' is similar with that of ',' and '.'. The minor difference is that results of MKMS is somewhat better than that of MLSVM. In the experiment of '/' and ')', DeepSVM successes in the highest accuracy, and MKMS's performance is superior to that of MLSVM and SVM. In the experiment of '!' and '?', MKMS obtains the best performance among all

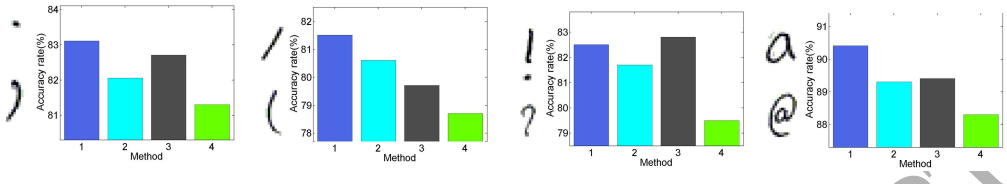


Figure 4: The results on ABCDETC dataset. 1: DeepSVM, 2: MLSVM, 3: MKMS, 4: SVM

algorithms, and the accuracy rate of DeepSVM is higher than that of MKMS and SVM.

Generally speaking, DeepSVM has boosted the accuracy rate with 1.3 percent, compared with that of MLSVM and MKMS, and with 2.8 percent compared to that of SVM. Meanwhile, MLSVM and MKMS' performance are very similar and close to each other. By the horizontal comparison, we can further confirm DeepSVM's performance to guarantee the effectiveness of Ex-Adaboost and deep structure.

5. Conclusion

In this paper, we first describe an Ex-Adaboost learning strategy, and then propose a new Deep Support Vector Machine (called DeepSVM) based on the general boosting procedure. All experiments demonstrated the robustness and effectiveness of the proposed method. In the meantime, these work clearly manifested that the deep structure and Ex-Adaboost learning strategy indeed offered a significant improvement for DeepSVM. On the other hand, deep learning is a new branch of machine learning, which has been proved to be a powerful tool of feature learning, and also SVM is regarded as one of the most powerful methods for classification and regression in machine learning community. Thus, how to combine their advantages to construct stronger classifier is an interesting and essential research topic. Based on these consideration, this paper carried out a simple of classification algorithm exploiting the deep structure and large margin principle. Admittedly, our current work is only a beginning about this topic. In the future, other classifiers will be employed, for example logistic regression, which can yield continuous output. In addition, we will evaluate our method on larger data sets, especially some complex image data set, based on various features, for example SIFT. Furthermore, we will prove the reasonable analysis in theoretical manner.

Particular, how can we explain the boost effect of deep structure based classifier?

Acknowledgment

This work has been partially supported by grants from National Natural Science Foundation of China (NO.61472390, NO.61402429, NO.11271361), key project of National Natural Science Foundation of China (NO.71331005, NO.91546201), the CAS/SAFEA International Partnership Program for Creative Research Teams, Major International(Ragional) Joint Research Project (NO.71110107026).

References

Abdullah, A., Veltkamp, R. C., Wiering, M. A., 2009. An ensemble of deep support vector machines for image categorization. In: SOCPAR'09. pp. 301–306.

Arnold, L., Rebecchi, S., Chevallier, S., Paugam-Moisy, H., 2011. An introduction to deep learning. In: ESANN.

Azamathulla, H. M., Wu, F.-C., 2011. Support vector machine approach for longitudinal dispersion coefficients in natural streams. Applied Soft Computing 11 (2), 2902 – 2905.

Bengio, Y., 2009. Learning deep architectures for ai. Foundations and trends in Machine Learning 2 (1), 1–127.

Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade. Springer, pp. 437–478.

Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (8), 1798–1828.

Bourlard, H., Kamp, Y., 1988. Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics 59 (4-5), 291–294.

Carrasco, M., López, J., Maldonado, S., 2015. A multi-class svm approach based on the l 1-norm minimization of the distances between the reduced convex hulls. Pattern Recognition 48 (5), 1598–1607.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.

Cho, Y., Saul, L. K., 2009. Kernel methods for deep learning. In: NIPS. Vol. 9. pp. 342–350.

Cho, Y., Saul, L.-K., 2010. Large-margin classification in infinite neural networks. Neural Computation 22 (10), 2678–2697.

Ciresan, D. C., Meier, U., Gambardella, L. M., Schmidhuber, J., 2010. Deep, big, simple neural nets for handwritten digit recognition. Neural computation 22 (12), 3207–3220.

Czibula, G., Czibula, I. G., Gaceanu, R. D., 2014. A support vector machine model for intelligent selection of data representations. Applied Soft Computing 18 (0), 70 – 81.

Deng, N., Tian, Y., 2009. Support vector machines: Theory, Algorithms and Extensions. Science Press, China.

Ekici, S., 2012. Support vector machines for classification and locating faults on transmission lines. Applied Soft Computing 12 (6), 1650 – 1658.

Freund, Y., Schapire, R., 1999. A short introduction to boosting. Japonese Society for Artificial Intelligence 14 (5), 771–780.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36 (4), 193–202.

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., Ng, A. Y., 2009. Measuring invariances in deep networks. In: Advances in neural information processing systems. pp. 646–654.

Hinton, G. E., 1989. Connectionist learning procedures. Artificial intelligence 40 (1), 185–234.

Hinton, G. E., 2009. Deep belief networks. Scholarpedia 4 (5), 5947.

Hinton, G. E., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. Neural computation 18 (7), 1527–1554.

Hinton, G. E., Salakhutdinov, R. R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.

Hinton, G. E., Zemel, R. S., 1994. Autoencoders, minimum description length, and helmholtz free energy. Advances in neural information processing systems, 3–3.

Japkowicz, N., Hanson, S. J., Gluck, M. A., 2000. Nonlinear autoassociation is not equivalent to pca. Neural computation 12 (3), 531–545.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y., 2009. What is the best multi-stage architecture for object recognition? In: Computer Vision, 2009 IEEE 12th International Conference on. IEEE, pp. 2146–2153.

Kim, S., Kavuri, S., Lee, M., 2013. Deep network with support vector machines. In: Neural Information Processing. Springer, pp. 458–465.

Kim, S., Yu, Z., Kil, R. M., Lee, M., 2015. Deep learning of support vector machines with class probability output networks. Neural Networks 64, 19–28.

Le Cun, B. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1990. Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems. Citeseer.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324.

Murphy, P. M., Aha., D. W., 1992. UCI machine learning repository.

N, S. R., Deka, P. C., 2014. Support vector machine applications in the field of hydrology: A review. Applied Soft Computing 19 (0), 372 – 386.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1988. Learning representations by back-propagating errors. MIT Press, Cambridge, MA, USA.

Schwenk, H., Milgram, M., 1995. Transformation invariant autoassociation with application to handwritten character recognition. Advances in neural information processing systems, 991–998.

Simard, P., Steinkraus, D., Platt, J. C., 2003. Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR. Vol. 3. pp. 958–962.

Socher, R., Huang, E. H., Pennin, J., Manning, C. D., Ng, A. Y., 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems. pp. 801–809.

Stigler, S. M., 1989. Francis galton's account of the invention of correlation. Statistical Science 4(2), 73–79.

Tang, Y., 2013. Deep learning using linear support vector machines. In: Workshop on Challenges in Representation Learning, ICML.

Tian, Y., Qi, Z., Ju, X., Shi, Y., Liu, X., 2013. Nonparallel support vector machines for pattern classification. IEEE transactions on Systems, Man and Cybernetics-PartB.

Vapnik, 1995. The nature of statistical learning theory. Springer-Verlag New York, Inc.

Vapnik, V., 2006. Estimation of Dependences Based on Empirical Data (Information Science and Statistics). Springer.

Weston, J., Collobert, R., Sinz, F., Bottou, L., Vapnik, V., 2006. Inference with the Universum. In: ICML '06. pp. 1009–1016.

Wiering, M., Schutten, M., Millea, A., Meijster, A., Schomaker, L., 2013a. Deep support vector machines for regression problems. In: Proceedings of the International Workshop on Advances in Regularization, Optimization, Kernel Methods, and Support Vector Machines: theory and applications.

Wiering, M., Van der Ree, M., Embrechts, M., Stollenga, M., Meijster, A., Nolte, A., Schomaker, L., 2013b. The neural support vector machine. In: Proceedings of the 25th Benelux Artificial Intelligence Conference (BNAIC).